Bureaux d'études en traitement des images

ENSEIRB, Filière Télécommunications, 3ème année, Option ISNC
ENSEIRB, Filière Electronique, 3ème année, Option TSI

ANNEE 2017-2018

M. DONIAS

PARTIE N° 1 REDRESSEMENT

Dans cette partie, nous nous proposons de « redresser » l'image *implant.bmp* représentant un implant dentaire observé par radiographie. La technique comporte trois étapes : un seuillage, une estimation de l'orientation de l'implant et un redressement.

1.1 Seuillage

Un seuil peut être déterminé automatiquement par la mise en œuvre de l'algorithme de classification « K-means ». Cet algorithme est adapté car l'histogramme de l'image traitée se compose de trois classes distinctes correspondant respectivement :

1. au fond,
2. à la gencive,
3. à l'implant.

Soient trois classes notées $L_i \left(i \in [1,3]\right)$ et caractérisées par un représentant noté $b_i \left(i \in [1,3]\right)$. Les $M \times N$ pixels (échantillons) de l'image, où M et N sont respectivement le nombre de lignes et de colonnes, sont répartis au sein des trois classes au moyen d'un critère de proximité ou de dispersion J qui s'exprime par

$$J = \sum_{i=1}^{3} \sum_{x_l \in L_i} d(x_l, b_i)$$

où $d(\bullet,\bullet)$ est un opérateur de distance et x_l représente l'intensité du pixel à la position l. On montre que le critère J est minimal si le représentant de chaque classe correspond à son barycentre :

$$b_i = \frac{1}{N_i} \sum_{x_l \in L_i} x_l$$

où N_i est le nombre de pixels inclus dans la classe L_i.

L'algorithme itératif du calcul des classes et de leur barycentre est mis en oeuvre comme suit :

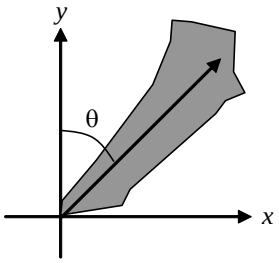
1. On choisit, par exemple au hasard, 3 représentants initiaux $\left\{b_1^{[0]}, b_2^{[0]}, b_3^{[0]}\right\}$ de classes (l'exposant i représente l'itération i).
2. A l'itération p, on associe chaque échantillon x_l à la classe $L_i^{[p]}$ si $d(x_l, b_i^{[p]}) < d(x_l, b_j^{[p]}) \quad \forall j \neq i$.
3. Les nouveaux barycentres $\left\{b_1^{[p]}, b_2^{[p]}, b_3^{[p]}\right\}$ de chaque classe $L_i^{[p]}$ sont recalculés selon $b_i^{[p]} = \frac{1}{N_i} \sum_{x_l \in L_i^{[p]}} x_l \quad \forall i$.

4. L'algorithme est réitéré depuis l'étape 2 tant que les centres des classes à l'itération *p* sont différents de ceux obtenus à l'itération précédente.

Proposer une initialisation particulièrement adaptée au problème. Visualiser l'image des indices de classe (étiquettes) après convergence de l'algorithme. A partir des barycentres obtenus, proposer une valeur de seuillage. Calculer et visualiser l'image binarisée (par convention, on attribuera la valeur 1 aux pixels appartenant à l'implant et la valeur 0 aux autres).

1.2 Estimation de l'orientation de l'implant

Afin de redresser l'implant, il est nécessaire d'estimer l'angle de rotation de l'implant par rapport à la verticale. Parmi les différentes techniques existantes, nous allons utiliser une méthode basée sur la diagonalisation de la matrice de covariance C du nuage de points formé des pixels appartenant à l'implant (valeur 1).



Orientation du vecteur propre associé à la plus grande valeur propre.

A partir des coordonnées *x* et *y* des pixels de l'implant, la matrice C se calcule selon :

$$C = \begin{bmatrix} C_{xx} & C_{yx} \\ C_{xy} & C_{yy} \end{bmatrix} \quad \text{où} \quad \begin{cases} C_{xx} = \dfrac{1}{N}\sum_{k=1}^{N}(x_k - \bar{x})^2 \\ C_{yy} = \dfrac{1}{N}\sum_{k=1}^{N}(y_k - \bar{y})^2 \\ C_{yx} = C_{xy} = \dfrac{1}{N}\sum_{k=1}^{N}(x_k - \bar{x})(y_k - \bar{y}) \end{cases} \quad \text{et} \quad \begin{cases} \bar{x} = \dfrac{1}{N}\sum_{k=1}^{N} x_k \\ \bar{y} = \dfrac{1}{N}\sum_{k=1}^{N} y_k \end{cases}.$$

Calculer les vecteurs propres v_1 et v_2 de la matrice de covariance C ainsi que les valeurs propres associées λ_1 et λ_2.

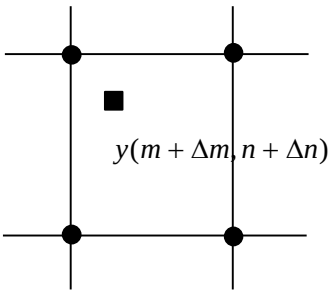
Calculer l'angle qui existe entre la direction verticale et le vecteur propre associé à la plus grande valeur propre.

1.3 Redressement

La transformation géométrique qui doit être appliquée à l'image traitée est une rotation dont l'angle a été estimé dans la question précédente. Afin de l'implémenter, il est possible d'utiliser une technique d'interpolation telle que celle qui est présentée ci-après.

Soit à estimer la valeur

$$f(m+\Delta m, n+\Delta n) \quad \text{où} \quad \Delta m \text{ et } \Delta n \in [0,1].$$



Des approximations au premier ordre de dérivés partielles fournissent

$$\begin{cases} f(m+\Delta m, n+\Delta n) \approx f(m+\Delta m, n) + \Delta n \dfrac{\partial f(m+\Delta m, n)}{\partial n} \\ \dfrac{\partial f(m+\Delta m, n)}{\partial n} \approx f(m+\Delta m, n+1) - f(m+\Delta m, n) \end{cases}$$

$$\begin{cases} f(m+\Delta m, n+1) \approx f(m, n+1) + \Delta m \dfrac{\partial f(m, n+1)}{\partial m} \\ \dfrac{\partial f(m, n+1)}{\partial m} \approx f(m+1, n+1) - f(m, n+1) \end{cases}$$

$$\begin{cases} f(m+\Delta m, n) \approx f(m, n) + \Delta m \dfrac{\partial f(m, n)}{\partial m} \\ \dfrac{\partial f(m, n)}{\partial m} \approx f(m+1, n) - f(m, n) \end{cases}.$$

En utilisant ces différentes relations, il apparait l'expression finale

$$\begin{aligned} f(m+\Delta m, n+\Delta n) \quad = \quad & (1-\Delta m)(1-\Delta n) f(m, n) \\ & + \Delta m(1-\Delta n) f(m+1, n) \\ & + \Delta n(1-\Delta m) f(m, n+1) \\ & + \Delta n \Delta m f(m+1, n+1) \end{aligned}$$

qui est connue sous le nom d'interpolation bilinéaire.

Proposer une implémentation du redressement sous *Matlab* à l'aide de la fonction **interp2** en décrivant la manière dont l'interpolation est mise en oeuvre dans le cadre d'une transformation géométrique.

PARTIE N° 2 SEGMENTATION SPATIO-TEMPORELLE

L'objectif de cette partie est de localiser des objets en mouvement (véhicules et piétons) dans une séquence vidéo au moyen de deux méthodes.

2.1 Détection par différence

2.1.1 Principe

Une première technique, implémentée dans de nombreux systèmes de vidéo-surveillance, s'appuie sur une image de référence. Celle-ci ne doit pas contenir d'objets en mouvement. La détection s'obtient par une simple différence d'intensité d'une image à la suivante, calculée point par point. Ceci convient, par exemple, pour un système d'alarme à condition de réactualiser régulièrement l'image de référence afin de prendre en compte l'évolution naturelle de la luminance de la scène observée.

2.1.2 Travail à réaliser

L'image de référence, dans cet exercice, est *ref1.bmp* tandis que les images représentant les objets en mouvement sont *im0.bmp*, *im1.bmp* et *im2.bmp*. Implémenter un schéma calculant une différence simple. Comparer les résultats avec un schéma basé sur une moyenne des différences calculée sur des blocs 16×16 partitionnant l'image sans recouvrement (fonction **blkproc**).

2.2 Calcul du flot optique

Dans un objectif de caractérisation plus fine, des informations telles que la direction et l'amplitude du mouvement sont des primitives intéressantes. La technique à implémenter ici consiste à estimer un champ de déplacement c'est-à-dire un vecteur déplacement pour chaque pixel de l'image. Les hypothèses sont, d'une part, que les variations de la luminance ne sont dues qu'au mouvement des objets et, d'autre part, que la luminance est constante le long du déplacement des objets. Ceci se traduit par l'équation

$$I(x, y, t) = I(x + d_x, y + d_y, t + T_e)$$

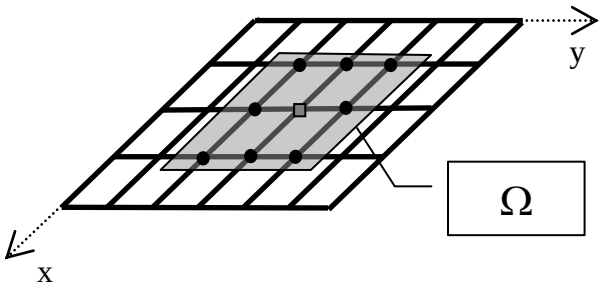
où (d_x, d_y) représente le vecteur de déplacement et T_e l'écart temporel entre deux images consécutives. En calculant le développement de Taylor au premier ordre du terme de droite, nous obtenons :

$$d_x \frac{\partial I(x,y,t)}{\partial x} + d_y \frac{\partial I(x,y,t)}{\partial y} + Te \frac{\partial I(x,y,t)}{\partial t} = 0 .$$

Cette équation, appelée équation de contrainte de mouvement, est à l'origine de nombreuses méthodes d'estimation du champ de déplacement.

2.2.1 Obtention d'une solution à l'équation de contrainte de mouvement

Afin d'obtenir une estimation « robuste » du flot optique, il est nécessaire de faire une hypothèse supplémentaire : le vecteur déplacement est supposé constant sur une fenêtre d'observation notée Ω.



L'équation initiale de contrainte du mouvement n'est alors plus vérifiée en tout point et devient :

$$d_x \frac{\partial I(x,y,t)}{\partial x} + d_y \frac{\partial I(x,y,t)}{\partial y} + Te \frac{\partial I(x,y,t)}{\partial t} = e(x,y,t)$$

où *e* désigne une erreur au modèle de mouvement. En minimisant l'erreur quadratique sur le voisinage Ω, établir l'expression du déplacement.

2.2.2 Estimation des dérivées partielles

L'évaluation du flot optique nécessite le calcul de dérivées partielles spatiales qui peuvent être estimées avec les filtres de Sobel :

$$S_x = \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \text{ et } S_y = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

2.2.3 Travail à réaliser

1. Visualiser, pour chaque pixel, la valeur absolue de la différence simple entre les images *taxi15.tif* et *taxi16.tif*. Commenter.

2. Ecrire une fonction qui réalise l'estimation des dérivées partielles à l'aide des filtres de Sobel (fonction **conv2** avec le paramètre ***'same'***). Visualiser notamment le module du gradient spatial.

3. Implémenter un programme permettant de calculer le vecteur de déplacement pour une région centrée sur un pixel d'une image. Le voisinage bidimensionnel d'estimation du flot optique $\Omega_{x,y} = \{x + [-L:L], y + [-L:L]\}$ sera paramétré par la variable L variant de 2 à 15. Visualiser les vecteurs de déplacement associés à plusieurs points obtenus à partir de la fonction **ginput**. Conclure.

4. Calculer le flot optique pour l'image complète. Le voisinage sera déterminé par L=15 et le champ de déplacement pourra être visualisé en s'inspirant de l'extrait de code suivant :

```
[m,n,l]=size(A);
figure, image(uint8(A)), colormap(gray(256), hold on
N=10;
[X,Y]=meshgrid(1:N:n,1:N:m);
quiver(X,Y,dx(1:N:m,1:N:n),dy(1:N:m,1:N:n),'r')
```

5. Trouver une valeur de flot optique (seuil) donnant le masque des objets en mouvement.

PARTIE N° 3 FLOT OPTIQUE MULTI-ECHELLE

Le flot optique est classiquement estimé en minimisant une équation différentielle au sens des moindres carrés. Plusieurs approximations (un développement de Taylor d'ordre un et une hypothèse de valeur constante sur un voisinage) sont nécessaires pour parvenir à une formulation analytique des composantes du flot optique. La taille limitée du voisinage pris en considération dans les calculs, par exemple 3×3 pour estimer les différentes dérivées premières partielles, conditionne la valeur maximale du déplacement estimé. L'implémentation d'un schéma multi-résolution, c'est à dire pyramidal, permet de s'affranchir de cette limitation. Les mouvements de forte amplitude peuvent en effet être détectés dans les niveaux supérieurs de la pyramide et propagés vers la base. Il devient alors possible d'estimer des déplacements plus importants que dans le cas d'une stratégie basée sur une seule échelle.

3.1 Compensation du mouvement

Rappelons que l'équation de flot optique s'écrit $I(x, y, t) = I(x + d_x, y + d_y, t + T_e)$.

Pour vérifier que la flot estimé est valide, il suffit de compenser le déplacement calculé sur l'image en *t*. Si l'estimation est correcte, l'image compensée doit en effet être identique à l'image en $t+T_e$. Sous *Matlab*, la compensation du mouvement peut être réalisée à l'aide de la fonction **griddata** qui n'impose pas que la grille d'entrée soit monotone.

Calculer le flot optique (carte de déplacement) pour les images *taxi15.tif* et *taxi16.tif* pour un voisinage carré de taille impaire dont la demi-taille est égale à *L=20*. A l'aide des fonctions **meshgrid** et **griddata**, compenser le déplacement calculé sur l'image *taxi15.tif*. Comparer l'image compensée à l'image *taxi16.tif* et commenter.

3.2 Construction d'une pyramide gaussienne

Visualiser les images couleur *rocade0.bmp* et *rocade1.bmp* qui sont extraites d'une séquence vidéo faisant apparaître un véhicule qui circule sur la rocade de Bordeaux et dont le déplacement est approximativement égal à 6 pixels/image.

Seule la luminance, obtenue par moyennage des composantes rouge, verte et bleue sera prise en compte.
Construire les deux pyramides gaussiennes (une pour chaque image). Afficher les différentes images et commenter.

3.3 Implémentation de l'algorithme multi-échelle

L'apport de l'approche multi-échelle est de calculer les mouvements (de forte amplitude) dans les niveaux supérieurs de la pyramide et de les propager vers les niveaux inférieurs. La compensation de mouvement permet de prendre en compte les déplacements déjà calculés.

Les étapes à réaliser pour chaque niveau de la pyramide sont :

1. Interpolation (**interp2)** du mouvement en provenance du niveau supérieur (sauf dans le premier niveau). Prendre garde à ce que la carte de déplacement du niveau supérieur ait une taille deux fois plus petite que celle du niveau courant. Il faut également prendre en considération le fait qu'un déplacement de 1 pixel/image dans le niveau supérieur est équivalent à un déplacement de 2 pixels/image dans le niveau courant.
2. Compensation du mouvement sur la première image. Réaliser l'opération de compensation sur la première image à l'aide de la carte de déplacement qui vient d'être calculée.
3. Estimation du déplacement par rapport à l'image compensée. Il est ici nécessaire de calculer à nouveau les dérivées premières partielles (spatiales et temporelle).
4. Actualisation de la carte de déplacement. Cumuler le déplacement calculé pour le niveau courant avec celui des niveaux précédents.

L'algorithme prend fin lorsque le dernier niveau de la pyramide est traité.

Implémenter l'algorithme multi-échelle pour le calcul du flot optique et l'appliquer sur les images *rocade0.bmp* et *rocade1.bmp*. Utiliser une fenêtre d'estimation de demi-taille L=5 pour tous les niveaux. Comparer la compensation de la première image à celle de la deuxième. Commenter les résultats pour plusieurs profondeurs de pyramide (1 à 4 niveaux).